UNIVERSIDAD DE LA RIOJA

TRABAJO FIN DE ESTUDIOS

Título

WebApp de un Explorador Genómico

Autor/es

JAVIER AYALA PALACIOS

Director/es

JUAN FÉLIX SAN JUAN DÍAZ y ROSARIO LÓPEZ GÓMEZ ,

Facultad

Facultad de Ciencia y Tecnología

Titulación

Grado en Ingeniería Informática

Departamento

MATEMÁTICAS Y COMPUTACIÓN

Curso académico

2016-17

publicaciones.unirioja.es
E-mail: publicaciones@unirioja.es

Facultad de Ciencia y Tecnología

Departamento de Matemáticas y Computación

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

WebApp de un Explorador Genómico

Autor:

Javier Ayala Palacios

Tutores:

Rosario López Gómez

Juan Félix San Juan Díaz

Logroño, Junio de 2017

Resumen

En la actualidad, los modelos SaaS (Software as a Service) que son ofrecidos a través de un navegador web están en completo auge. Una de las razones más sencillas y evidentes de este hecho es que permiten ofrecer un servicio de forma sencilla para cualquier usuario con nociones de ofimática básicas, puesto que todo el proceso se desarrolla desde un entorno básico y conocido. Otra razón, quizás no tan notoria para una persona sin los conocimientos adecuados, pero igualmente fácil de asimilar, es la posibilidad de ofrecer un software muy potente, sin la necesidad de un hardware a la misma altura, que no todo el mundo tiene, ni se puede permitir.

En este proyecto se presenta un modelo SaaS aplicado al campo de la Bioinformática. Esta aplicación permitirá a un investigador, a través de un navegador web, hacer uso de un *software de exploración genómica* que requiere de grandes recursos de computación, supliendo con ello las carencias en prestaciones que un ordenador normal posee.

Summary

Nowadays, SaaS models (Software as a Service), which are offered through a web explorer, are growing. One of the simplest and most obvious reasons of this fact is, that they can offer a service in an easy way for anyone with basic knowledge in computing, since all the process is developed in a basic environment. Another reason, maybe not so noticeable for common people, but equally easy to understand, is the chance to offer a very powerful software, without having a capable hardware, which not everybody can have or afford.

In this project, a SaaS model is shown, applied to bioinformatics. This app will make a researcher be able to use a genomic explorer software by a web explorer, which need big computation resources.

Agradecimientos

En primer lugar, agradecer a mis tutores, tanto Rosario como Juan Félix, que me hayan ayudado a la consecución de este proyecto, unas veces haciendo de guía, para que pudiera conseguir llegar a la orilla sin desviarme, otras aguantándome, cada vez que tenía algo que enseñar con mis avances.

Agradecer a los investigadores del *CIBIR* el tiempo que les he quitado para mejorar aspectos de la aplicación desarrollada, y los buenos ratos que me han hecho pasar durante el tiempo libre, ya que me he sentido uno más allí.

También quiero dar las gracias a mis compañeros de carrera por amenizarme el paso por esta Universidad, así como por ayudarme cuando tenía dudas de carácter más técnico. Me llevo, no sólo conocimiento, sino también grandes amigos.

Por último, agradecer a mi familia y amigos su apoyo incondicional en momentos buenos y no tan buenos, no sólo durante la realización de este Trabajo Final de Grado, sino durante toda la carrera, este soporte me ha permitido llegar hasta el final.

Índice

1. Introducción

1.1. Contexto

En un tema tan complejo como es la genómica, es interesante comenzar con unas definiciones básicas, que ayuden posteriormente a una mayor comprensión del contexto de este proyecto:

- **Genoma de un individuo:** Conjunto completo de su información genética, en el que se incluyen sus propios genes, reguladores y zonas intergénicas.
- **Genoma de una especie:** Genoma o patrón de referencia, que normalmente no se corresponde con un único individuo, sino que para su secuenciación se ha utilizado ADN de múltiples individuos. Por ejemplo, el genoma (de referencia) humano se trata de una secuencia mosaico de ADN de numerosos donantes anónimos, entre los que se encuentran individuos de distintas etnias, tanto hombres, como mujeres.

El genoma humano contiene 3000 Mb (MegaBase) distribuidas en 23 cromosomas. 1 Mb equivale a 10^6 pb (Par de Bases Nitrogenadas, la unidad de medida del ADN). Esto significa que ha sido necesario secuenciar, ordenar y solapar correctamente, como mínimo, unos cuatro millones de fragmentos para obtener toda la secuencia genómica.

Los primeros métodos de secuenciación rápida se desarrollaron a finales de la década de 1970. Gracias al avance en distintos campos hoy día el proceso es mucho más rápido, simple y automatizado, ya que diversas empresas han desarrollado distintas plataformas de secuenciación, conocidas como *Next-Generation Sequencing*.

Cuando se secuencia el ADN de un individuo se pueden encontrar variantes de un único nucleótido entre el genoma de referencia y los datos obtenidos. En el caso de que dicha variante concreta esté presente en, al menos, un 1% de la población, se considerará un *polimorfismo*. Estas variaciones no son otra cosa que mutaciones que hacen variar a la especie en un intento por evolucionar. En el caso de que se alteren unos pocos nucleótidos se consideraría una *mutación génica*, por el contrario, si afecta a grandes fragmentos del ADN, se conoce como *mutaciones cromosomáticas*.

Pasando a un punto de vista más informático, una distribución concreta de un genoma estará compuesta por la secuencia de nucleótidos de cada cromosoma, además de una serie de anotaciones pertinentes a las distintas características que se encontraron dentro de dicha secuencia. Cuando se analiza la secuencia de una región cromosomática, el bioinformático es capaz de acceder a dicha información a través de un programa especial llamado navegador genómico.

Un **navegador genómico** es una aplicación informática que proporciona un conjunto de herramientas que permiten realizar una exploración eficiente a lo largo de una secuencia genómica, así como de las anotaciones cartografiadas en su interior.

Dentro de un navegador genómico existirán varios puntos de entrada para acceder a la información de un elemento en concreto, y dependerá de los datos que conozcamos la elección del método más adecuado para su búsqueda.

1.2. Motivaciones

Los investigadores del *Centro de Investigación Biomédica de La Rioja* (CIBIR) precisan de un software de exploración genómica, como el descrito en el apartado anterior, para poder avanzar en la realización de sus trabajos diarios. Sin embargo, el uso de *navegadores genómicos* por parte de un usuario sin experiencia presenta varios problemas:

- **Instalación complicada:** El primer gran problema al que se enfrentará cualquier trabajador que pretenda instalar un navegador genómico es la complejidad en su instalación. Existen programas de este tipo en el que dicha instalación no es trivial, ya que se necesita validar una licencia que depende de tu dirección IP y del nombre de tu máquina, en un tiempo muy limitado. Además, hay programas que requieren, para un total funcionamiento, de la obligatoriedad de una distribución en particular del sistema operativo utilizado, como por ejemplo, *Windows 7 Enterprise,* distribución que no es muy común en los ordenadores de gama media que suelen manejar los investigadores. Para ser aún más precisos, varios de los empleados tuvieron un cursillo de instalación y uso para este software en concreto, en el que nadie consiguió, ni siquiera el primer paso, es decir, instalar la aplicación, a pesar de ser incluso una aplicación de pago.
- **Software complejo:** En el raro caso de tener un ordenador ejecutando el programa, un investigador se encontrará con otro gran problema, y es que la aplicación es muy completa, tanto, que el usuario se pierde en la gran

cantidad de funcionalidad que ofrece, sin ser capaz de realizar lo que realmente necesita.

- **Recursos de computación elevados:** Aún con todo, la máquina que debería ejecutar el software tiene que ser de altas prestaciones. Existen productos de pago que ofrecen, además de un visualizador genómico, diversas herramientas para la realización de análisis de secuenciación masiva, pero tienen el inconveniente de que, para poder aprovechar todas las ventajas que el software ofrece se necesitan potentes equipos de cálculo. Estos equipos sólo se encuentran en la Plataforma de Genómica y Bioinformática del CIBIR, con recursos hardware de 24 núcleos de ejecución y 128GB de memoria principal, y no están disponibles para cualquier investigador. Puntualizar que, a pesar de las características del hardware, durante una sesión de uso intensivo, se puede llegar a usar hasta la partición de intercambio para realizar parte de sus cómputos. Para el uso de programas con la única funcionalidad de *navegador genómico* se necesita fundamentalmente capacidad de almacenamiento y bastante memoria RAM.

Los tres problemas descritos anteriormente son los que han motivado el desarrollo de este proyecto, que permitirá a los investigadores realizar su trabajo de una manera más sencilla y óptima.

Otro de los motivos que han influenciado el desarrollo de este proyecto ha sido el deseo de mejorar la manera de trabajar del investigador. Normalmente un investigador analiza y estudia los resultados de unos ficheros donde se le informan de las variantes encontradas e información de las anotaciones de las mismas de la muestra analizada respecto a un genoma de referencia. Por otro lado, utiliza un navegador genómico para visualizar las secuencias analizadas y estudiar en detalle algunas de las variantes; ello requiere la introducción manual en el programa de la localización exacta de la variante para que el navegador genómico se desplace a esa posición. Con este proyecto no sólo se ha construido un navegador genómico al uso, que permita la visualización de secuencias del genoma secuenciado, sino que además se ha integrado un sistema de visualización del fichero de variantes donde, tan sólo pulsando en cada una de ellas, el navegador genómico se desplace a su localización, de una manera automática y transparente para el usuario. Se ofrece, de esta forma, una mayor interacción y rapidez por parte del investigador en el estudio de los resultados.

1.3. Objetivo

La solución a los problemas anteriormente descritos para los investigadores que necesitan el uso de este tipo de herramienta pasa por la creación de una aplicación web que permita hacer uso de un navegador genómico, donde no se requieren del resto de utilidades que otros softwares comerciales añaden. Además es un plus añadido el poder disponer de una aplicación que no suponga coste alguno para la organización y que no requiera capacidad de cómputo y altas prestaciones en los equipos de los investigadores. Todo esto lo hemos conseguido creando una aplicación web que de respuesta a las necesidades mencionando e integrando en ella un visualizador web genómico haciendo uso de una API de programación proporcionada por la empresa Broad Institute, totalmente gratuita.

Dicha API de programación ha sido desarrollada mediante lenguaje *JavaScript* y manejará únicamente los recursos específicos para visualizar el contenido necesario, dejando a un servidor que ejecute los cálculos que sean requeridos. Con esto se resolverían los 3 problemas iniciales:

- **No existiría instalación**, puesto que todo funcionaría en un servidor donde se aloja la aplicación, accesible a través de un navegador web cualquiera.
- **El software dejará de ser complejo**, puesto que la librería está adaptada y optimizada para funcionar en un navegador, con únicamente las funcionalidades necesarias para dar respuesta a las necesidades de sus usuarios.
- Directamente ligado con la afirmación anterior, **no se necesitará un hardware potente en el lado del cliente.** Se consigue con ello, por tanto, un mayor acceso a este recurso por parte de cualquier investigador, ya no que hay dependencia importante de recursos por parte del mismo para su uso. En el mundo de la informática y la programación, muchas veces todo se basa en un *compromiso* que debes optimizar para adecuarlo a tu situación. En este caso en concreto, aceptamos el compromiso de tener una funcionalidad reducida (ya que nos es suficiente con ella), a cambio de no necesitar equipos muy bien dotados de hardware (que son muy caros) y de evitarnos los problemas de instalar el software en muchas máquinas distintas.

Por tanto, el objetivo del proyecto consistirá en construir una aplicación web que permita usar un explorador genómico vía web. Además, deberá ser integrada dentro de un sitio web de mayor tamaño, y que también ofrece otras funcionalidades dedicadas a los investigadores del *CIBIR*, como son el acceso a la localización exacta de las variantes/ a estudiar en dicho navegador genómico, así como otras funcionalidades de filtrado, exportación, captura de imágenes, etc… de las variantes y área de interés.

2. Estudio de Viabilidad del Proyecto

Antes de dar comienzo al proyecto, se comprobó que efectivamente era viable su realización desde todos los aspectos:

- **Tecnología:** El software debería ser una aplicación web accesible fácilmente a través de un navegador, en la que se pudiera no sólo manejar de forma sencilla una tabla de variantes sino además visualizar esta información de manera gráfica a través de un navegador genómico. Por tanto, una condición indispensable sería que desde la propia tabla se pudiera, de manera interactiva, actuar en el navegador genómico, para visualizar gráficamente la información seleccionada, facilitando así al investigador la ardua tarea de revisión y desplazamiento manual en el navegador genómico hasta la posición de búsqueda deseada.
    - Para la tabla de variantes se propuso usar la librería para lenguaje de programación PHP llamada *MATE* (*MySQLAjaxTableEditor*), conocida por la tutora y usada en otros proyectos en el CIBIR.
    - Para el navegador genómico, la misma empresa que desarrolla el programa que actualmente usan los investigadores, ofrece otra librería de lenguaje JavaScript llamada *IGV.JS*, con la que se realizaron las pruebas pertinentes y se comprobó que su uso y automatización eran posibles.
- **Tiempo:** Tras poco trabajo se pudo llegar a un prototipo muy cercano a lo que sería la aplicación final, que estaría dotada de una funcionalidad ligeramente superior y con todo el proceso automatizado.
    - Esto fue debido a que se comprobó “a mano” que el prototipo era usable mediante una web generada de forma totalmente estática. Ya que el tiempo necesario para la realización de esta primera aproximación fue muy reducido, se consideró que sería suficiente con el tiempo del que se disponía para su correcto desarrollo y puesta en funcionamiento.

Sabiendo que la tecnología era conocida y fácil de usar, **el proyecto se consideró viable** y se pudo continuar con su desarrollo.

Además, se realizó un pequeño estudio de alternativas para el navegador genómico, con el fin de manejar la mayor cantidad de opciones y poder elegir la mejor de todas ellas. Sin embargo, no se encontró ninguno.

3. Planificación

3.1. Alcance del Producto

Como ya se ha explicado anteriormente, el producto que va a desarrollarse consiste en la implementación de una aplicación web que emule el comportamiento de un explorador genómico en un navegador. Con él se pretende que los investigadores del *CIBIR* puedan realizar su trabajo de una forma más sencilla y eficiente. El desarrollo se realizará a través de varias librerías que permitirán su óptimo funcionamiento, a través de los lenguajes de programación PHP y JavaScript.

El producto no estará compuesto únicamente por el software construido, sino que, además, en el alcance del producto también se incluye su integración en un sitio web más grande (ya elaborado), que también aporta diversas funcionalidades a los citados investigadores. En concreto, dicha integración se realizará en un sistema construido con un *CMS* (Content Management System, *Sistema de Gestión de Contenidos*) **Drupal** (`Versión 7.53`).

Por último, se considerará también alcance del producto la gestión y el procesado de los diferentes archivos con los que se nutrirá la aplicación, y que tendrán que ser subidos al servidor por parte de un usuario para su posterior uso en la aplicación web.

3.2. Metodología elegida

Puesto que antes de comenzar el presente proyecto se realizó un estudio de viabilidad mediante un estudio, se decidió seguir una **metodología de desarrollo en cascada**.

Esta decisión se tomó en base a que durante esta temprana fase se tenían bien definidos los requisitos del proyecto completo, con lo cual el primer paso sería fácil de realizar y favorecería un desarrollo rápido y de calidad.

3.3. Entregables del Producto

Una vez finalizado el proyecto, se obtendrán los siguientes entregables:

- **Código Fuente:** Todo el código desarrollado, con su correspondiente documentación. Puesto que será integrado en un CMS, la estructura que

podría tener el código (en cuanto a la localización de los archivos) sigue la estructura facilitada por Drupal.

- **Máquina Virtual:** La aplicación web se encuentra dentro de un sistema virtualizado con sistema operativo Centos, versión 7, y éste alojado también en un servidor Centos 7 con gran potencia de cálculo.
- **Memoria del Proyecto:** Documento en el que se explican todos los aspectos del proyecto que se va a desarrollar.

3.4. Gestión del Tiempo

La gestión del tiempo se realizará mediante la creación de la EDT (*Estructura de Descomposición del Trabajo*), que permitirá definir cada una de las actividades a realizar durante la realización del proyecto. Tras esto podrá establecerse el marco temporal completo, con lo que se podrán generar los diagramas de hitos y Gantt del proyecto.

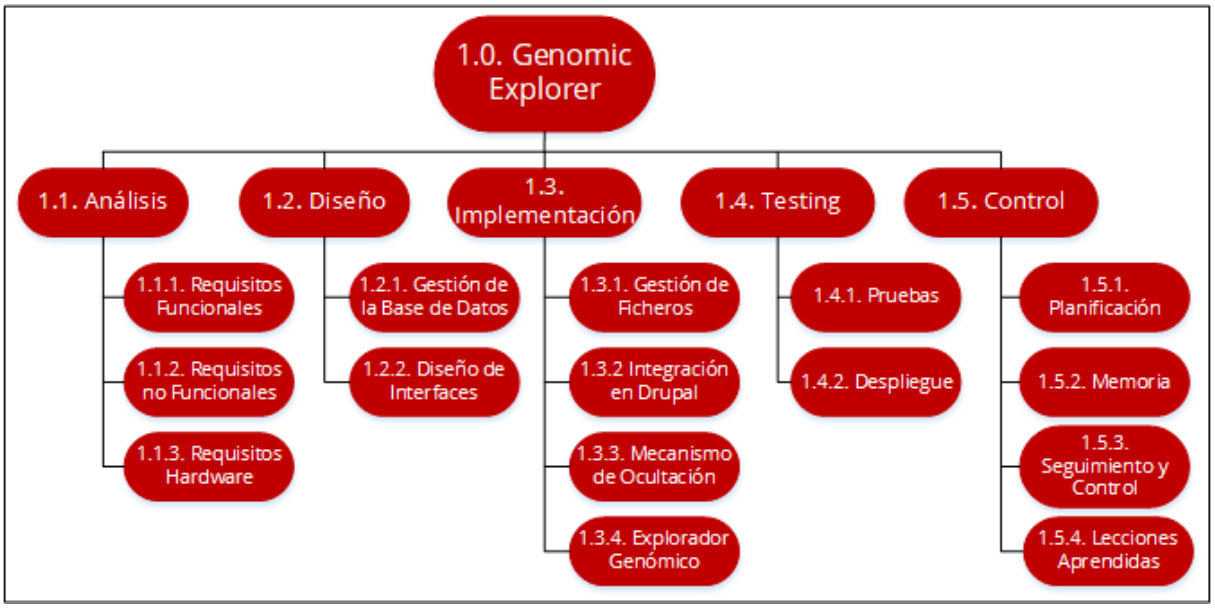

Diagrama 1. Estructura de Descomposición del Trabajo (EDT).

A continuación, se realiza una breve descripción de cada una de las actividades a realizar durante el desarrollo del proyecto definidas en la EDT (*Diagrama 1*):

- **(1.1.1) Requisitos Funcionales:** Definir los distintos requisitos funcionales con los que debe cumplir el producto.
- **(1.1.2) Requisitos no Funcionales:** Definir los distintos requisitos no funcionales con los que debe cumplir el producto.
- **(1.1.3) Requisitos Hardware:** Definir los requisitos hardware que debe ofrecer la máquina que ejecute el producto.

- **(1.2.1) Gestión de la Base de Datos:** Al estar dentro de un CMS, no se diseñará una base de datos desde cero, sino que se usará la del propio sistema gestor de contenido. En esta tarea se deberá estudiar la base de datos para poder usarla correctamente en fases posteriores del desarrollo. A pesar de este hecho, sí que se manejarán tablas propias, que provienen del fichero CSV, este fichero es el archivo que subirá el usuario con el listado de variantes de su muestra. Cada usuario dispondrá de una tabla (ya que no se comparten los datos, ni están relacionadas entre sí).
- **(1.2.2) Diseño de Interfaces:** Diseño de prototipos de interfaces para su posterior implementación.
- **(1.3.1) Gestión de Ficheros:** Implementación de un mecanismo que permita volver a usar los ficheros una vez estén subidos al servidor por parte del usuario, ya que son de gran tamaño y costaría una notable cantidad de pérdida de tiempo y de recursos de computación volver a hacer esto de nuevo. Además, al ocupar tanto espacio, también se encuentra dentro de esta tarea el borrado posterior de los archivos, una vez pasado un tiempo (que se decidirá en fases posteriores). Por último, la aplicación deberá tener cierto grado de tolerancia a fallos debido, de nuevo, a la naturaleza de los ficheros.
- **(1.3.2) Integración en Drupal:** Crear dentro del CMS un mecanismo que permita tanto la gestión de los ficheros, como el uso del explorador genómico.
- **(1.3.2) Explorador Genómico:** Construcción del software que permite usar un explorador genómico, con las características ya descritas, vía web.
- **(1.3.4) Mecanismo de Ocultación:** Aunque la aplicación web esté integrada con Drupal y funcionará a través del CMS, el explorador genómico diseñado estará fuera del entorno de Drupal. El explorador genómico diseñado está compuesto por una tabla de variantes y un navegador genómico. En dicha tabla se debe mostrar información de la variante anotada (una serie de campos relativos a bases de datos que expliquen la variante), por tanto aprovechar el tamaño completo de la web es esencial, ya que aumenta la productividad drásticamente. El CMS limita el espacio a un 75%, por lo que esta parte concreta del software estará “fuera” de él. No se puede, por tanto, realizar un aprovechamiento de los mecanismos de seguridad que Drupal ofrece, por lo que se ideará un sistema completo de ocultación que evitará posibles ataques externos.

- **(1.4.1) Pruebas:** Pruebas reales realizadas en la aplicación.
- **(1.4.2) Despliegue:** Integración de la máquina virtual en el sistema de producción. Comprobación y verificación de su correcto funcionamiento.
- **(1.5.1) Planificación**: Elaboración de un plan de trabajo que se especificará en la memoria, y en el que se refleje la distribución temporal de las tareas.
- **(1.5.2) Memoria:** Redacción de la memoria completa en la que se deben describir completamente todos los aspectos y procesos que han tenido lugar durante el desarrollo del proyecto.
- **(1.5.3) Seguimiento y Control:** Durante todo el proyecto deberá seguirse un control de todo lo realizado, y ser contrastado junto con la planificación para comprobar el progreso total del proyecto.
- **(1.5.4) Lecciones Aprendidas:** Redacción en la memoria de aquellos aspectos más prácticos, importantes o de interés que se han aprendido durante la realización del proyecto y que puedan ser valiosos de cara al futuro.

Código	Nombre	Estimación (horas)
1.1.1	Requisitos Funcionales	10
1.1.2	Requisitos no Funcionales	10
1.1.3	Requisitos Hardware	5
1.2.1	Gestión de la Base de Datos	15
1.2.2	Diseño de Interfaces	10
1.3.1	Gestión de Ficheros	25
1.3.2	Integración en Drupal	40
1.3.3	Mecanismo de Ocultación	25
1.3.4	Explorador Genómico	40
1.4.1	Pruebas	10
1.4.2	Despliegue	10
1.5.1	Planificación	20
1.5.2	Memoria	50
1.5.3	Seguimiento y Control	20
1.5.4	Lecciones Aprendidas	10
Tiempo total estimado:		**300**

Tabla 1. Relación entre las actividades y sus estimaciones temporales.

Tras describir todas las actividades, se presenta una tabla (*Tabla 1*), en la que se puede ver el tiempo estimado para cada una de ellas, además de su código dentro del proyecto y su nombre.

Como se puede ver, la estimación total es de 300 horas, que son las asignadas a un proyecto de estas características. El total, deberá repartirse en 18 semanas de trabajo, que son las comprendidas entre los días 13 de Febrero de 2017 y 23 de Junio de 2017, excluyendo fines de semana, días festivos según el calendario académico de la Universidad de La Rioja, del curso 2016/17 y la semana de vacaciones de Semana Santa 2017 (del 17 al 23 de Abril). Durante las dos semanas previas al comienzo del proyecto, se comprobó la viabilidad del mismo ya que si éste no lo era, se manejarían otras alternativas. En concreto se dispondrá de 84 días de trabajo, en los que se estima trabajar entre 3:30-4 horas durante cada jornada.

Semana	Inicio	Fin	Observaciones
1	13/02/2017	19/02/2017	
2	20/02/2017	26/02/2017	
3	27/02/2017	5/03/2017	
4	6/03/2017	12/03/2017	
5	13/03/2017	19/03/2017	
6	20/03/2017	26/03/2017	
7	27/03/2017	2/04/2017	
8	3/04/2017	9/04/2017	
9	10/04/2017	16/04/2017	Los días 13 y 14 son festivos.
-	17/04/2017	23/04/2017	Vacaciones de Semana Santa.
10	24/04/2017	30/04/2017	
11	1/05/2017	7/05/2017	El día 1 es festivo.
12	8/05/2017	14/05/2017	El día 12 es festivo.
13	15/05/2017	21/05/2017	
14	22/05/2017	28/05/2017	
15	29/05/2017	4/06/2017	
16	5/06/2017	11/06/2017	El día 9 es festivo.
17	12/06/2017	18/06/2017	El día 12 es festivo.
18	19/06/2017	23/06/2017	

Tabla 2. Semanas de trabajo.

Con el marco temporal totalmente establecido, podemos fijar los hitos a conseguir para la consecución del proyecto (*Diagramas 2 y 3*):

Hito \ Semana	1	2	3	4	5	6	7	8	9
Planificación		♦							
Análisis de Requisitos				♦					
Diseño				♦					
Implementación									♦
Integración en Drupal									
Pruebas reales									
Entrega de Proyecto									

Diagrama 2. Diagrama de Hitos (#1).

Hito \ Semana	10	11	12	13	14	15	16	17	18
Planificación									
Análisis de Requisitos									
Diseño									
Implementación									
Integración en Drupal				♦					
Pruebas reales						♦			
Entrega de Proyecto									♦

Diagrama 3. Diagrama de Hitos (#2).

Febrero 2017												
Semana	1					2					3	
Tarea \ Día	13	14	15	16	17	20	21	22	23	24	27	28
1.1.1												
1.1.2												
1.1.3												
1.2.1												
1.2.2												
1.3.1												
1.3.2												
1.3.3												
1.3.4												
1.4.1												
1.4.2												
1.5.1												
1.5.2												
1.5.3												
1.5.4												

Diagrama 4. Diagrama de Gantt (Febrero).

Marzo 2017																					
Semana	3			4					5					6					7				
Tarea \ Día	1	2	3	6	7	8	9	10	13	14	15	16	17	20	21	22	23	24	27	28	29	30	31
1.1.1																							
1.1.2																							
1.1.3																							
1.2.1																							
1.2.2																							
1.3.1																							
1.3.2																							
1.3.3																							
1.3.4																							
1.4.1																							
1.4.2																							
1.5.1																							
1.5.2																							
1.5.3																							
1.5.4																							

Diagrama 5. Diagrama de Gantt (Marzo).

Abril 2017												
Semana	8					9			10			
Tarea \ Día	3	4	5	6	7	10	11	12	24	25	26	27	28
1.1.1													
1.1.2													
1.1.3													
1.2.1													
1.2.2													
1.3.1													
1.3.2													
1.3.3													
1.3.4													
1.4.1													
1.4.2													
1.5.1													
1.5.2													
1.5.3													
1.5.4													

Diagrama 6. Diagrama de Gantt (Abril).

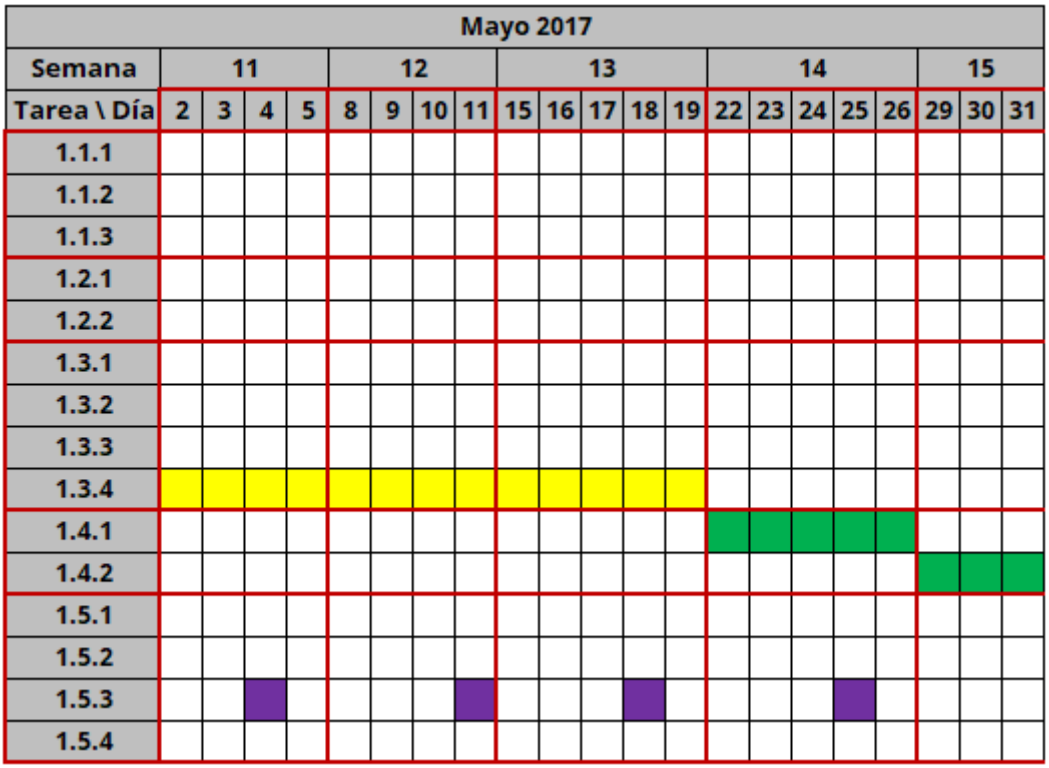

Diagrama 7. Diagrama de Gantt (Mayo).

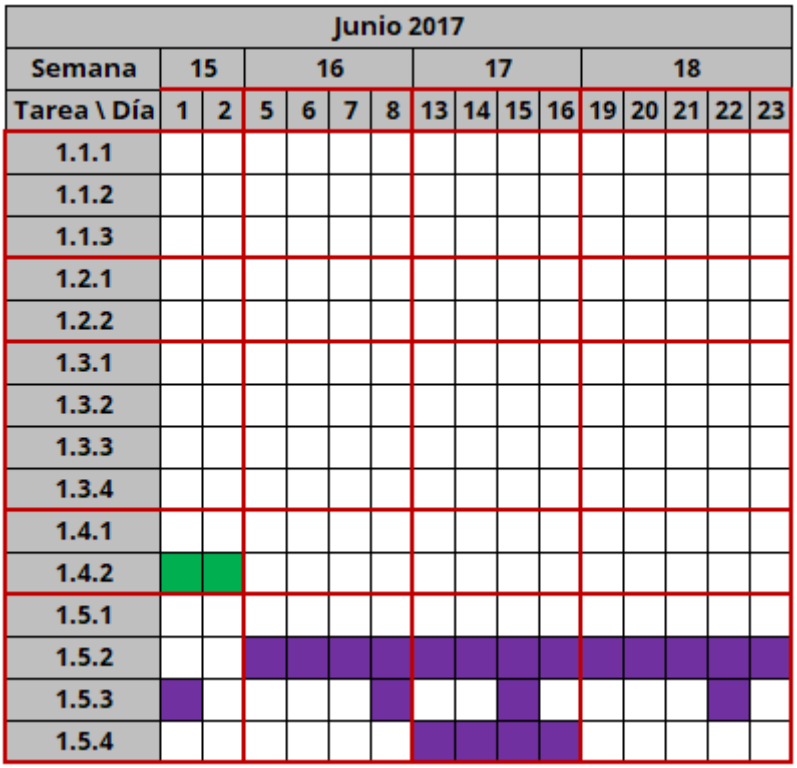

Diagrama 8. Diagrama de Gantt (Junio).

Finalmente, se muestra el diagrama de Gantt del proyecto (*Diagramas 4, 5, 6, 7 y 8*) en el que se muestran los tiempos en los que se realizará cada tarea del proyecto, agrupado por meses.

Para entender los sucesivos diagramas de Gantt cabe mencionar que se comienza con la planificación, se sigue con el ciclo de vida en cascada habitual, para terminar con la redacción de la memoria y las lecciones aprendidas. Por último, el seguimiento y control se realizará cada jueves, ya que es el día consensuado con la tutora de empresa para comprobar los avances y preparar la siguiente semana.

3.5. Plan de Adquisiciones

Las adquisiciones que tendrán que realizarse para la realización de este proyecto serán las siguientes:

- **IGV.js:** Librería gratuita y *Open Source*, que permite el uso de un navegador genómico a través de lenguaje JavaScript. El proyecto se encuentra en *GitHub* y pertenece a la misma empresa que suministra otro software que los investigadores del CIBIR pueden instalarse el local, aunque éste debe hacerse en un computador que disponga de ciertos recursos importantes.
    - Enlace al proyecto en *GitHub*: https://goo.gl/1DyrRT
- **MATE:** Librería gratuita y de pago, que permite la gestión completa de tablas de una base de datos de forma ágil y sencilla para el usuario, usando para ello la tecnología *AJAX*, como su nombre completo indica (*MySQLAJAXTableEditor*).
    - Página oficial de *MATE*: https://goo.gl/UhtcyD
- **Drupal:** Sistema gestor de contenido libre, modular y multipropósito, totalmente configurable. Además, es completamente *Open Source* y muy seguro.
    - Página oficial de *Drupal*: https://goo.gl/tpIJZd
- **Plantilla Business:** Plantilla para la interfaz de Drupal con diseño `web responsive`.
    - Página oficial de *Business:* https://goo.gl/mCjtlT
- **Centos 7:** Sistema operativo para servidores gratuito y de código abierto, que deriva de la distribución *Red Hat*. Se trata de un sistema robusto y estable, además de tener facilidad tanto para su instalación, como para su uso.
    - Página oficial de *Centos*: https://goo.gl/o1lDw2
- **VirtualBox:** Programa de virtualización gratuito que albergará la máquina virtual con la web completa con el sistema diseñado.
    - Página oficial de *VirtualBox*: https://goo.gl/n2rbY2

- **MySQLServer:** Motor para la base de datos del servidor, caracterizado por su rapidez.
    - Página oficial de *MySQLServer*: https://goo.gl/BCkIQK
- **PhpMyAdmin:** Herramienta para la gestión de la base de datos MySQL.
    - Página oficial de *PhpMyAdmin*: https://goo.gl/TQ1NBZ
- **Contenido multimedia:** Se usarán contenido e imágenes provenientes del *CIBIR* para la presentación del software dentro de la web, otorgándole un aspecto más profesional.

4. Desarrollo del Proyecto

4.1. Análisis de Requisitos

4.1.1. Requisitos Funcionales

Los requisitos funcionales que deberá cumplir la aplicación desarrollada serán los descritos a continuación:

- **Gestión de usuarios:**
  - Cualquier usuario podrá solicitar una cuenta a través de un formulario web.
  - El administrador podrá dar de alta o no (aceptar/rechazar una cuenta), cambiar sus permisos/roles, activar/desactivar y eliminar las distintas cuentas dentro del sistema.
  - Una vez el usuario tenga una cuenta aceptada y activada, pasará a ser un usuario registrado, con lo que podrá autenticarse en el sistema y gestionar los datos personales que haya introducido.
- **Gestión de ficheros de datos genómicos:**
  - Un usuario registrado podrá subir ficheros de distintos tipos (CSV, BAM, BAI, VCF e IDX), para que puedan ser guardados en el servidor. Antes de proceder con esta subida, se realizarán ciertas comprobaciones de los archivos para asegurarse que serán compatibles con la aplicación.
  - El usuario registrado podrá comprobar los ficheros que tiene actualmente en el servidor para acceder de nuevo a la aplicación con ellos.
  - Los archivos subidos serán eliminados según una política de borrado que será definida en fases posteriores del desarrollo.
- **Visualización de datos genómicos:**
  - A partir del archivo CSV se generará una tabla de variantes genómicas, en la que se podrá ver la información relativa a cada una. Por otro lado se podrán realizar búsquedas (tanto simples como avanzadas) y mostrar/ocultar información.
  - Además, esta tabla incluirá un botón que permitirá localizar la variante concreta dentro del visualizador genómico, una funcionalidad esencial para los investigadores.

4.1.2. Requisitos No Funcionales

Por otra parte, la aplicación también deberá cumplir con los siguientes requisitos no funcionales:

- La aplicación deberá responder de forma rápida a las interacciones con el usuario.
- El idioma de la aplicación será el inglés.
- Será tolerante ante posibles fallos de procesamiento en los archivos.

4.1.3. Requisitos Hardware

Puesto que es una aplicación en la que se depende de un servidor externo que es el que nos proporciona los recursos necesarios, habrá que distinguir entre la parte cliente, y la del servidor:

- **Cliente:**
  - El servicio que ofrecería una aplicación de estas características normalmente requeriría de un hardware muy potente, ya que se trabaja con un volumen de datos muy alto, que no cualquier máquina es capaz de manejar. Es por ello que se construye esta aplicación apoyándose en un servidor, de manera que desde el lado del cliente tan sólo se necesite un navegador web actualizado, que permita el uso de cookies y JavaScript.
  - Conexión a Internet con suficiente velocidad para poder subir y bajar los datos con el servidor.
- **Servidor:**
  - La aplicación se nutre de archivos de gran tamaño (del orden de gigabytes), por lo que se necesitará una gran capacidad de almacenamiento.
  - Conexión a Internet de suficiente velocidad para poder subir y bajar los datos con el cliente.
  - Servidor web capaz de interpretar lenguaje PHP y usar el mecanismo CORS.

4.1.4. Diagramas de Casos de Uso

Tras haber especificado los requisitos, se van a distinguir 3 actores dentro del sistema (visitante, usuario y administrador), con lo que podemos obtener los siguientes diagramas de casos de uso:

- **Visitante:** Se entenderá por visitante a cualquier persona que visite la aplicación y que pueda solicitar una cuenta para ingresar dentro del sistema.

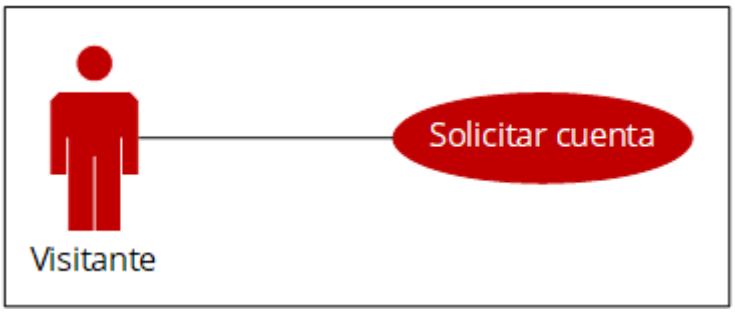

Diagrama 9. Casos de uso del actor Visitante.

  - *Solicitar cuenta:* Proceso por el que cualquier persona realiza una petición para crear una cuenta en el sistema y así acceder a los servicios que ofrece.
- **Usuario:** Cuando un visitante obtenga una cuenta, se convertirá en usuario registrado en la aplicación, por lo que podrá acceder a ella para usar toda la funcionalidad asociada a su rol.

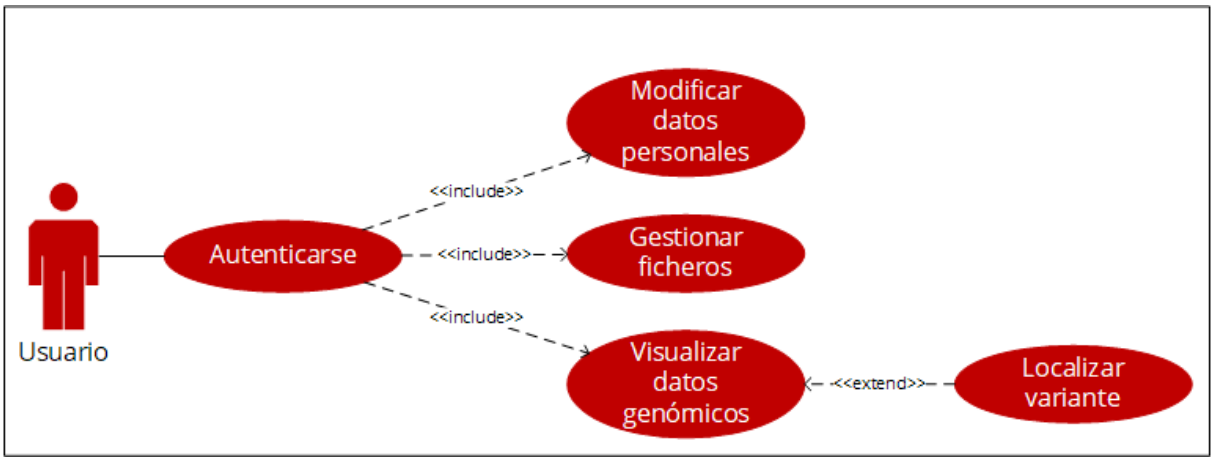

Diagrama 10. Casos de uso del actor Usuario.

  - *Autenticarse:* El usuario introduce sus credenciales para poder ingresar en la aplicación.

- *Modificar datos personales:* Añadir, modificar o eliminar los datos personales asociados a su cuenta en el sistema.
- *Gestionar ficheros:* Se podrá ver el listado completo de los ficheros (asociados al usuario) que estén almacenados en ese momento en el servidor para poder acceder a la aplicación con ellos nuevamente.
- *Visualizar datos genómicos:* Parte primordial de la aplicación que permitirá acceder al explorador genómico para realizar labores de investigación.
- *Localizar variante:* Funcionalidad esencial que permitirá seleccionar una fila de la tabla de variantes para ser localizada dentro del visualizador genómico.

- **Administrador:** Será quien gestione todo lo relacionado con las cuentas del resto de los usuarios del sistema.

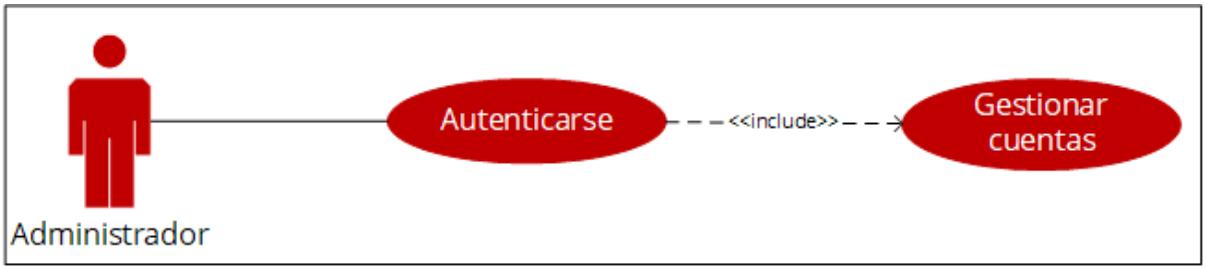

Diagrama 11. Casos de uso del actor Administrador.

- *Autenticarse:* Proceso idéntico al del actor Usuario, pero asociado esta vez al Administrador.
- *Gestionar cuentas:* Permitirá aceptar/rechazar solicitudes de cuentas, además de activar/desactivar, cambiar roles/permisos o eliminar dichas cuentas.

4.2. Diseño

4.2.1. Gestión de la Base de Datos

En lo que respecta a la base de datos, tenemos dos aspectos a tener en cuenta. El primero de ellos es que no será necesario crear una base de datos desde cero, puesto que el CMS Drupal gestiona toda su estructura y contenido a través de una. El segundo, es la necesidad de cada usuario de tener su propia tabla de variantes, en la que se guarden los datos de la misma.

Estudio de la base de datos de Drupal

Drupal es un CMS muy caracterizado por guardar una cantidad muy alta de su información en su base de datos. Un ejemplo de este hecho es que existen muy pocos archivos con HTML para la formación de sus páginas, y esto se debe a que este código fuente es guardado directamente dentro de la base de datos. Cuando los necesita, accede a ellos.

En nuestro caso, de la base de datos completa, sólo nos interesa la parte en la que se almacenen todos los datos generados a partir de una subida de archivos por parte del usuario. Esto Drupal lo gestiona mediante una tabla fija, y otra dinámica que se genera cuando un nuevo tipo de archivo se sube:

- **Tabla general (`file_managed`):** Es la tabla general en la que se almacenan los datos referidos a los archivos que se han subido y Drupal conoce (identificador del fichero, identificador del usuario que lo subió, nombre del archivo, URL relativa al directorio público del servidor, tipo MIME, tamaño, fecha de subida y tipo de archivo).
- **Tablas dinámicas (`field_data_<tipo_archivo>`):** Al diseñar un formulario que recoja los campos que serán mostrados al usuario para la subida de archivos el propio Drupal genera una tabla por cada tipo de archivo definido. Así cuando el usuario cumplimenta el formulario web y sube los archivos solicitados, determinada información será guardada en estas tablas (tipo de formulario que lo subió, identificador del formulario usado, saber si el archivo está borrado o no, identificador de subida e identificador del fichero).

Gracias a la relación dada en estas dos tablas (a través del identificador del fichero), se puede acceder al path donde queda almacenado cada fichero. Con esta ruta pública seremos capaces de usar la librería IGV.js para que acceda a los datos de dicho fichero y pueda usarlos.

Tablas de los usuarios

Una de las partes primordiales dentro de un explorador genómico es la tabla de variantes, que proviene de un archivo CSV. Gracias a los datos que contiene, un investigador puede acceder a una zona específica del navegador genómico relativos a una variante en particular, en la que existen datos interesantes para él.

Es por ello que cada usuario necesitará una tabla en la base de datos y se poblará con la información contenida en ese archivo subido. Seguidamente se cargará una parte de la web mostrando en una tabla HTML un cierto número de variantes, y permitiendo al usuario a través de un sistema de acceso y filtrado el que pueda acceder a la visualización del resto de variantes almacenadas en la tabla de la base de datos. La estructura para dicha tabla es fija, ya que tiene que responder al tipo de información y formato que el archivo CSV tiene establecido. Estos ficheros CSV, con la información de las mutaciones anotadas en una muestra clínica, han sido generados a través de un protocolo de secuenciación y análisis dentro de la Plataforma de Genómica del CIBIR.

Desde un punto de vista purista de las bases de datos, tener varias tablas iguales en estructura, y que sólo se diferencian porque su nombre hará alusión al del usuario, es muy ilógico. Sin embargo, es una cuestión de eficiencia.

Podría optarse por usar una única tabla, a la que añadir un campo con el identificador del usuario para cada fila que se inserte desde el CSV que suba. Sin embargo, esto acarrearía problemas de acceso y posiblemente bloqueos continuamente. Las razones principales son las siguientes:

- **Al subir e insertar un nuevo fichero:** Se tendrían que borrar los datos que ya se tienen e insertar los nuevos.
    - *Tabla única:* Vaciar esa parte concreta de la tabla, sin tocar las demás. Los usuarios que estén usando la aplicación en ese momento deberán esperar si necesitan hacer otra petición a la tabla ya que está en uso.
    - *Varias tablas:* Vaciar la tabla del usuario que hizo la subida. Como cada tabla es independiente, el resto de usuarios no sufren este problema.
- **Al borrar un usuario:** Borrar sus datos.
    - *Tabla única:* El mismo problema anterior, hay que borrar sus datos, lo que repercutiría en el buen funcionamiento de la aplicación por parte de otros usuarios.
    - *Varias tablas:* Borrar su tabla concreta.

Es por esta razón por la que durante el diseño se ha optado por la opción de tener una tabla para cada usuario, de tal forma que el rendimiento de la base de datos

no se resienta al tener que realizar consultas continuamente a la misma tabla de la base de datos, sino que esté manejando recursos distintos para cada usuario.

Con lo cual, durante la creación de un nuevo usuario, se creará automáticamente también una tabla dentro de la base de datos que haga referencia al usuario para que puedan insertarse los datos. De igual manera, esta tabla será borrada cuando se elimine la cuenta de usuario.

4.2.2. Diseño de Interfaces

La aplicación principal estará compuesta por el explorador genómico, sin embargo, también tendrá otras dos interfaces, que son las referidas a cargar los últimos archivos aún en el servidor para su uso y también una página de errores en el procesado de los archivos que permita elegir la acción a realizar, de tal forma que sea tolerante a dichos errores. A continuación, se muestran los prototipos diseñados para las distintas interfaces (*Figuras 1, 2 y 3*), de manera que sirvan de referencia durante la implementación:

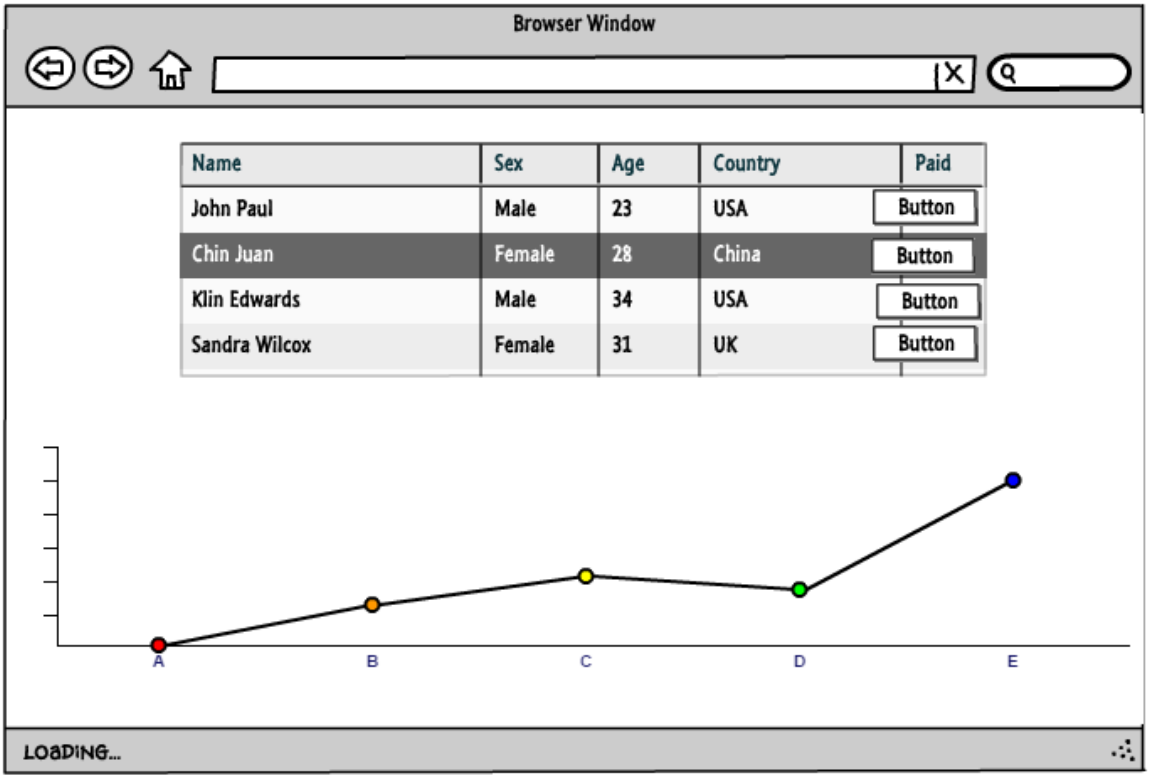

Figura 1. Prototipo de interfaz para el Explorador Genómico.

Esta interfaz está compuesta por la tabla de variantes (arriba) y el visualizador genómico (abajo). La tabla de variantes contiene un botón en cada fila, que es el que reposicionará el visualizador para que muestre gráficamente los datos.

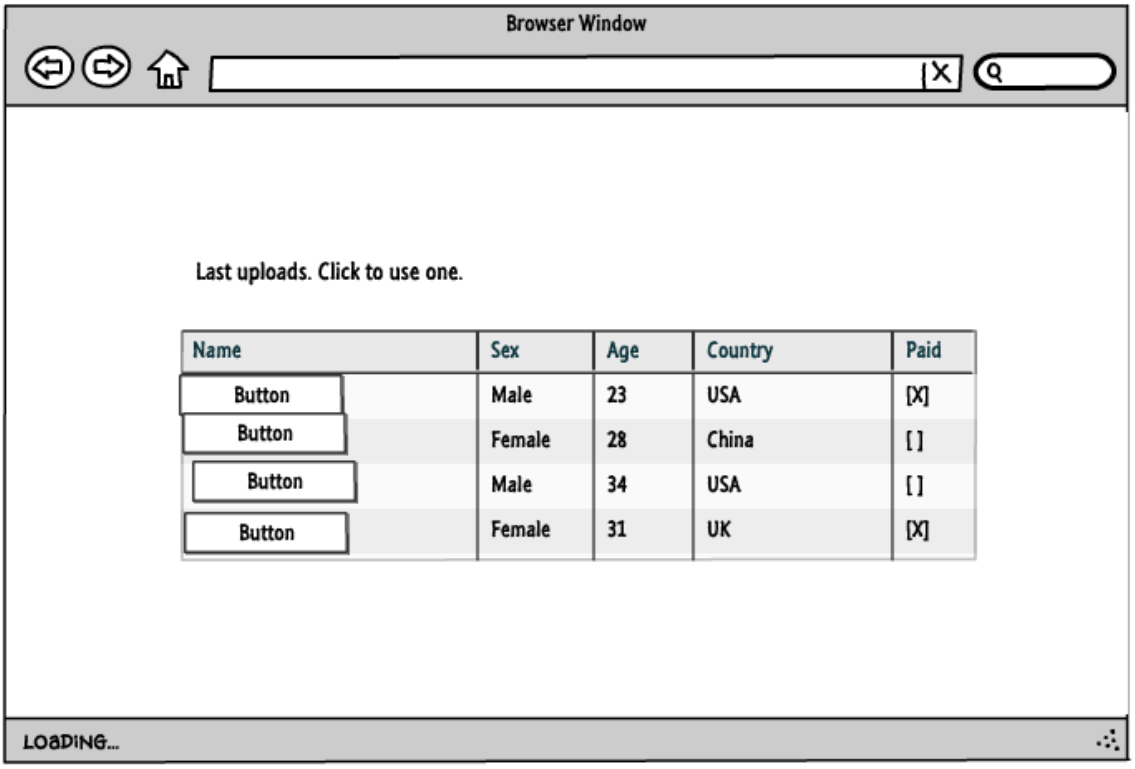

Figura 2. Prototipo de interfaz para las últimas subidas.

En esta interfaz se puede ver que en la tabla están los últimos conjuntos de subidas de archivos recogidos en una tabla, y al pulsar en uno de ellos podremos acceder al explorador genómico sin tener que realizar una nueva subida.

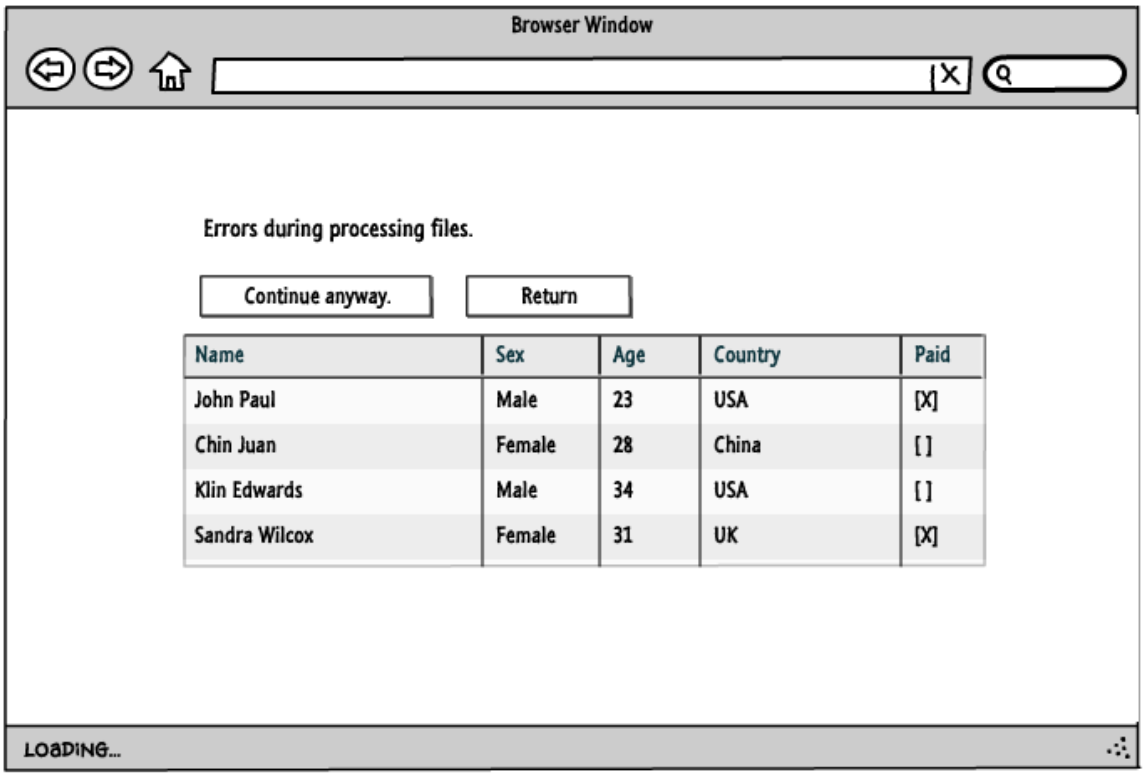

Figura 3. Prototipo de interfaz para los errores de procesamiento.

La última interfaz diseñada permite ver los errores de procesamiento de los archivos en una tabla (algunas variantes que no responden al patrón de almacenamiento en la tabla de la base de datos), y, tras controlar qué variantes no van a ser cargadas, nos permita continuar, o en caso contrario, volver a la página anterior.

La parte relativa al formulario que permite la subida de archivos se ha creado usando el entorno de Drupal, es por ello que no se ha diseñado una interfaz con código propio. Todos los prototipos han sido realizados mediante la herramienta de prototipado de interfaces llamada *Lumzy*:

- Web oficial de *Lumzy*: https://goo.gl/YqpWvq

4.3. Implementación

Durante todo este apartado se explicará por separado el funcionamiento de cada una de las partes que comprenden el software facilitando así su comprensión, y se terminará con la conjunción de ambos. El resultado final podrá ser visto en el *Anexo 1*.

4.3.1. Gestión de Ficheros

Subida de ficheros

La subida de ficheros al servidor será realizada por parte de Drupal, mediante un formulario simple. Se subirán 5 archivos: CSV, BAM, BAI, VCF e IDX, de los cuales, los tres primeros son obligatorios.

Uso de ficheros

Para que el explorador genómico pueda funcionar, sólo se necesita que los ficheros subidos tengan una URL pública, de tal manera que al referenciarlos puedan ser accedidos. Es por eso que serán guardados en una carpeta concreta del servidor al cual se pueda acceder.

Borrado de ficheros

Debido al tamaño de los ficheros, se ha estimado la necesidad de que sean borrados cuando hayan pasado 6 horas después de la subida. Esto se realizará mediante un script que el servidor ejecutará periódicamente, en el que listará la

carpeta en la que se encuentren los archivos subidos, comprobará si llevan más del tiempo establecido, y entonces procederá a su borrado.

4.3.2. Integración en Drupal

Tolerancia a errores

Los errores de procesado pueden ocurrir debido a errores de formato en el archivo CSV. Es el único archivo que debe procesarse, ya que se tiene que volcar íntegramente en la tabla de la base de datos correspondiente. Se ha desarrollado un pequeño modulo que controla los errores de inserción fila por fila, de tal manera que, se comienza a insertar todo el fichero y, si encuentra un error, lo guarda y continúa con la siguiente información. Una vez termine su procesado, pueden ocurrir dos cosas:

- **Existen errores de procesado:** Se muestra una pequeña página web, integrada en Drupal con los errores obtenidos, permitiendo al usuario continuar, o abortar. La opción de seguir de forma normal aunque existan fallos se debe a que los archivos tienen un tamaño enorme (pueden llegar a tener más de 100000 filas/variantes), y un único fallo de parseo de línea (o varios) normalmente no son tan graves como para dejar la aplicación inutilizable.
- **No existen errores de procesado:** Se redirecciona al usuario a la web del explorador genómico para permitirle al usuario interaccionar con el mismo y llevar a cabo el estudio de las variantes.

Últimas subidas

Desarrollo de un pequeño módulo integrado en Drupal, que permite al usuario ver las últimas subidas que ha hecho al servidor, de tal manera que no tenga que realizar el proceso de subida si quiere trabajar de nuevo sobre los mismos archivos.

No obstante, debido a la necesidad de almacenamiento final que el mantener estos ficheros requeriría a corto plazo, si el número de usuarios fuera elevado, se ha optado por la política de borrado de aquellos archivos que se encuentren almacenados en el servidor por un periodo superior a 6 horas. Este tiempo es el suficiente otorgado al investigador para realizar un análisis de las variantes de interés.

Salida al explorador genómico

Una vez que los ficheros han sido cargados y la información de las variantes almacenada en la base de datos será el usuario el que decida acceder al explorador genómico. En este momento nos salimos del entorno de Drupal para acceder a una nueva Web, fuera del entorno del propio CMS que permite visualizar a modo pantalla el explorador genómico y permite al usuario interaccionar con la información.

4.3.3. Mecanismo de Ocultación

Uno de los problemas de que el explorador genómico funcione fuera del entorno de Drupal es su posible vulnerabilidad, ya que no se disponen de los mismos mecanismos de autenticación y control que ofrece el CMS. La vulnerabilidad se encuentra asociada al hecho de que, con Drupal, cada vez que se cumplimenta un formulario de subida de archivos se almacena un identificador correlativo asociado al formulario cumplimentado. Es decir, se ha cumplimentado el formulario 123, luego el 124, y así seguidamente. La forma de construir inicialmente la web asociada al navegador genómico era siguiendo este patrón, por lo tanto, un atacante podría modificar en la URL el valor asociado al formulario y podría estar visualizando un navegador genómico con los datos almacenados de otro usuario. Si todo esto se hiciera dentro del entorno de Drupal sería el mismo Drupal quién impediría el acceso ya que el usuario autenticado no es el autor del formulario que pretende visualizar.

Por otro lado, necesitábamos ocupar el mayor tamaño posible de la pantalla por ello nos salimos de Drupal, por tanto, procedimos a crear un mecanismo de ocultación, que permitiera enmascarar los datos de una forma sencilla a través de un código aleatorio en cada subida. Este mecanismo funciona de la siguiente manera:

1. Al subir los ficheros, se crea dentro de una carpeta externa, otra carpeta que tiene de nombre un **código de 20 caracteres generado aleatoriamente**.
2. Dentro de la carpeta creada, se crea un archivo vacío que tiene por nombre un número, que es el identificador de subida del formulario, a través del cual podemos llamar a la base de datos en una consulta interna para obtener la URL pública de cada fichero.

3. El código aleatorio es pasado como parámetro vía *GET* al explorador genómico, de manera que puede buscar la ruta a la carpeta generada, y al fichero que contiene el número identificador de subida.
4. El explorador genómico sólo necesita de ese número para generar la página completa.

Con esto, los ficheros quedan ocultos ante posibles ataques. Es importante destacar que el código es alfanumérico, con una longitud de 20. Lo que equivale a que la posibilidad de que un usuario externo de justo con un valor aleatorio que se encuentre asociado a una carpeta creada en el servidor se reduce a:

$$7{,}4 \cdot 10^{-32}\% = 0{,}00000000000000000000000000000074\%$$

4.3.4. Explorador Genómico

El explorador genómico se implementará íntegramente mediante un único fichero PHP, que incluirá todo lo referido a las dos librerías necesarias, usadas conjuntamente.

Como ya ha sido explicado, el explorador genómico está compuesto mediante dos elementos fundamentales, los cuales son la tabla de variantes y el visualizador genómico. A su vez, estos se construyen de la siguiente manera:

- **Tabla de variantes:** Proviene de un archivo en formato CSV, que no es más que una extensión que indica que los datos que contiene están dispuestos en forma de tabla, como si la de una base de datos se tratara.
- **Visualizador genómico:** Se trata de una representación gráfica de una muestra de análisis genómico, contra un genoma de referencia, de tal forma que las diferencias entre ambos sean fácilmente identificables. El análisis genómico se realiza mediante un secuenciador de ADN, y genera varios tipos de archivos (BAM, BAI, VCF e IDX) con datos genómicos de distintos tipos en cada uno.
    - *Archivos BAM y BAI:* El archivo BAM almacena todas las secuencias de una muestra clínica que se ajustan de algún modo a la secuencia de ADN concreta del genoma de referencia sobre el que se están alineando. Debido al tamaño que suelen tener estos archivos, se usa el BAI, que no es más que un índice, que permite buscar de forma

más eficiente en el primero. Su especificación puede encontrarse en el siguiente enlace: https://goo.gl/tZzA4E.

- *Archivos VCF e IDX:* El archivo VCF contiene información completa sobre las variantes encontradas en la muestra clínica mencionada anteriormente. En esta información se encuentra también aquélla que resulta primordial para el navegador genómico que es la posición concreta en el genoma donde se observa la variante. De la misma manera que el anterior, el tamaño de los ficheros hace que el uso de un segundo fichero, el IDX, que funciona como índice, haga mucho más sencillo su acceso. Su especificación puede encontrarse en el siguiente enlace: https://goo.gl/HP6l4p.
- *Genomas de referencia:* Estos ficheros contienen la secuencia del genoma de referencia dentro de la especie y tiene variaciones en función de la versión del genoma utilizado. Por ejemplo, el genoma de referencia más utilizado actualmente para muestras de humano es el *hg19*.

Una vez conocemos los datos técnicos que necesitaremos, se puede comenzar a explicar cómo se va construir el elemento central de este proyecto. El explorador genómico tratará de una única página web en la que se podrá ver tanto la tabla de variantes, como el visualizador genómico, de tal manera que interactuando con la tabla se pueda reposicionar el visualizador para que se muestren los datos de interés para el investigador.

Para dotar de cierta estructura al software desarrollado en el proyecto, se creará una carpeta pública en el servidor, con la siguiente estructura:

```
GenomicExplorer
|-css
|-lib
|-media
|-php
|-resources
```

Figura 4. Estructura de la carpeta que albergará el Proyecto.

- **Carpeta `css`:** Hojas de estilo para darle un mejor aspecto al proyecto.
- **Carpeta `lib`:** Librerías que se utilizarán.
- **Carpeta `media`:** Recursos que se utilizarán para la presentación de la web.
- **Carpeta `php`:** Archivos PHP que generarán la página web.
- **Carpeta `resources`:** Recursos que albergará la web para su uso.

Tabla de variantes

La tabla de variantes será construida a través de la librería mostrada anteriormente, llamada *MATE*. Funciona mediante lenguaje PHP, y lo que permite es la gestión completa de tablas de una base de datos. Aunque tiene versiones de pago, tras su estudio, se llegó a la conclusión de que con la versión gratuita sería suficiente.

En el caso de este proyecto, lo que resulta útil es el uso de la tecnología *AJAX* para no tener que recargar la página cada vez que necesita obtener más datos, ya que las tablas que se visualizan contienen sólo un conjunto de variantes y no todo el contenido del fichero subido. El usuario, mediante un paginador dinámico, podrá desplazarse rápidamente entre conjuntos de variantes. Además, la web permite hacer filtrados de variantes y búsquedas entre sus campos, con filtros tanto sencillos como avanzados, lo cual será de gran utilidad para el investigador. Por último, también permite realizar exportaciones de los datos filtrados, en el caso de que un usuario así lo requiriese.

Tras haber descargado la librería, obtendremos un archivo comprimido. Una vez descomprimido, se obtiene el directorio llamado `MATE` que se alojará en `GenomicExplorer/lib` de nuestro proyecto.

El funcionamiento de la librería es muy sencillo, y abstrae al programador que la use de gestionar la conexión con la base de datos, ya que sólo precisa de los datos necesarios para conectarse. En concreto, se deberá acceder al fichero `GenomicExplorer/lib/MATE/DBC.php` (fichero que gestiona la conexión con la base de datos), y cambiar los datos de la conexión (*Código Fuente 1*), que se encuentran al comienzo del fichero dentro de la variable `$dbcInfo`.

```
private static $dbcInfo = array(
     'host' => '<HOST>',
     'db' => '<BASE_DE_DATOS>',
     'user' => '<USUARIO>',
     'password' => '<CONTRASEÑA>', // Esta coma es necesaria
);
```

Código Fuente 1. Conexión con la base de datos de *MATE*.

Tras esto, la librería ya está configurada para usarse en nuestra base de datos.

El siguiente paso es crear una página que muestre una tabla de la base de datos mediante la librería. El proceso es muy sencillo gracias a que la librería abstrae de muchos aspectos al programador, aun así, la documentación sirve de gran ayuda.

Para que MATE nos cree una tabla, el proceso debe realizarse mediante un script PHP que genere una página web con ciertas características, explicadas en estos tres pasos:

1. **Generación del objeto `AjaxTableEditor`:** Este objeto es el central de la librería, y lo que nos permite es darle todas las opciones necesarias para generar la tabla a nuestro gusto. Se realiza mediante código PHP. Además, en este paso es donde crearemos los botones permitan la interacción con el visualizador genómico.
2. **Generación de cabeceras HTML:** Una vez el PHP haya sido procesado, la web a la que haya dado lugar, debe contener las librerías necesarias para operar con la librería y la tecnología *AJAX* mediante JavaScript.
3. **Generación del contenedor de *MATE*:** Simplemente se necesitarán unos elementos `div` de HTML con una estructura concreta, en los que se insertará automáticamente todo el contenido a visualizar.

Visualizador genómico

El visualizador genómico se construye a partir de la librería *IGV.js* ya explicada, que, a su vez, se apoyará en los ficheros que haya subido el usuario al servidor para poder generar los datos correspondientes.

El funcionamiento es muy simple, tenemos que descargar la última versión del proyecto para descargar la librería. En nuestro caso, crearemos un directorio llamado `IGV` dentro de `GenomicExplorer/lib`, que será el que contenga los archivos necesarios para su correcto funcionamiento.

El proceso de uso de esta librería es bastante sencillo, ya que, al igual que anteriormente, sólo se necesita generar una página web con ciertas características:

1. **Cabeceras HTML:** De nuevo, se necesitarán incluir las cabeceras necesarias de la librería para que el software funcione correctamente, además de un pequeño código JavaScript en el que se configura mínimamente el

visualizador genómico, y se le dan las rutas a los ficheros que necesitará manejar

2. **Contenedor *IGV*:** Nuevamente es un único contenedor tipo `div`, que posteriormente será llenado por la librería.

Generación del Explorador Genómico

El resultado final de la implementación constará de un único fichero PHP, además de otro fichero de conexión a la base de datos en el que se apoyará para obtener todos los datos necesarios en la generación de la página web.

Uno de los mayores problemas iniciales durante la creación del prototipo ha sido el uso conjunto de las librerías mencionada, ya que ambas usan tecnología *AJAX* pero con librerías diferentes. La resolución final fue muy sencilla, ya que detectamos que la versión utilizada de *MATE* que inicialmente utilizamos funcionaba con *Prototype*, existiendo otra versión más actualizada que usaba *jQuery*, que es la misma tecnología que usa *IGV*, con lo que se resolvieron los problemas.

Con lo explicado anteriormente, se creó un fichero PHP que aunaba el uso de las dos tecnologías, y que permitía así, tras el procesado del código PHP en el servidor, el que se generara una página web que incluyera las funcionalidades de ambas librerías en una única, haciendo que interactuaran entre ellas.

El script creado no genera nada más que texto HTML, teniendo en cuenta los datos que se le proporcionan, con la siguiente estructura de código, para aumentar su legibilidad (*Código Fuente 2*):

```
<html>
     <head>
          <!-- CABECERAS MATE -->
          <!-- JAVASCRIPT MATE -->
          <!-- CABECERAS IGV -->
          <!-- JAVASCRIPT IGV -->
     </head>
     <body>
          <!-- CONTENEDOR MATE -->
          <!-- CONTENEDOR IGV -->
     </body>
</html>
```

Código Fuente 2. Estructura del HTML generado.

4.4. Testing

Durante el desarrollo de la aplicación se realizaron varias pruebas que confirmaban que el software funcionaba de la forma deseada en todos los escenarios posibles.

Además, el explorador genómico se ha creado un *modo DEBUG* que se puede activar desde el código, y que permite acceder a la visualización de las variables que se utilizan para generar la página completamente, aspecto que fue muy útil a la hora de realizar las pruebas de funcionamiento, ya que ahorra mucho tiempo de espera en cargas.

De hecho, los mecanismos tanto de error, como de visualización de las subidas recientes nacen a partir de las pruebas que se realizaron, ya que permitían realizar todo el proceso de desarrollo de una forma mucho más rápida, aunque finalmente fueron incluidas como características debido a su amplia utilidad.

4.5. Despliegue de la Aplicación

El despliegue de la aplicación se ha realizado de forma sencilla debido a que sólo se necesitaba instalar la máquina virtual en el servidor para que comenzara a funcionar, además de cambiar las contraseñas relativas a la gestión de la base de datos, puesto que son distintas las utilizadas en el entorno de desarrollo, que en el de producción.

Además, esto ha sido muy sencillo debido a la separación del código, con lo que no ha habido ningún problema, y la aplicación ha funcionado perfectamente.

Por último, decir que también han tenido que cambiarse unos ajustes mínimos en el servidor, de manera que sería capaz de manejar paquetes de red que usaran el mecanismo *CORS*. Tampoco ha supuesto ningún problema, ya que esto era un requisito de una de las librerías, y venía explicado perfectamente en su documentación como hacerlo.

Se ha incorporado además el protocolo de seguridad HTTPS con la instalación de un certificado firmado por una CA (*Autoridad Certificadora*) configurando la redirección al puerto 443 para que la información sea transmitida de forma cifrada.

5. Seguimiento y Control

5.1. Objetivos alcanzados

En la realización de este proyecto se ha conseguido alcanzar el objetivo primario, que es la implementación de una aplicación web que permita utilizar la funcionalidad de un explorador genómico a través de Internet, de tal forma que el investigador que lo use no tenga que lidiar con los problemas de instalación, así como la necesidad de recursos altos de computación.

Además, se ha conseguido integrar el sistema en un sistema web mayor con otras funcionalidades interesantes para los investigadores, construida mediante Drupal.

Finalmente se han ampliado las funcionalidades del navegador genómico permitiendo la visualización del listado de variantes y su posicionado automático en la parte del genoma que le corresponde, funcionalidad que un programa de instalación local no dipone.

La integración en general ha constado de tres partes, en primer lugar, la ocultación mediante una secuencia aleatoria de caracteres de la aplicación para evitar posibles ataques. En segundo, un mecanismo de tolerancia a fallos ante el procesamiento de los ficheros, que permita el uso de la aplicación, aunque existan dichos errores. Y, por último, un sistema de últimas subidas al servidor, que permita al usuario usar el software de nuevo sin tener que realizar la carga de nuevo, debido a que el tamaño de los ficheros es muy elevado.

5.2. Desviaciones de Tiempo

Aunque durante el total del proyecto no existe una gran desviación, sí que existen en las diferentes partes pequeñas penalizaciones debido a que la implementación ha sido más costosa que lo planificado en inicialmente. Sin embargo, todo queda compensado a lo largo del resto de tareas del proyecto cuyo esfuerzo estimado se ha optimizado al máximo quedando corregido así el desfase inicial.

En la siguiente tabla (*Tabla 3*) se muestra el seguimiento del tiempo invertido al proyecto durante la duración del mismo, en la que se pueden ver también las desviaciones con datos reales, que explican lo que se ha aclarado durante el párrafo anterior:

Tarea	Horas estimadas	Horas reales	Desviación (horas)
Tarea 1.1.1	10	12	+2
Tarea 1.1.2	10	10	0
Tarea 1.1.3	5	5	0
Tarea 1.2.1	15	10	-5
Tarea 1.2.2	10	10	0
Tarea 1.3.1	25	20	-5
Tarea 1.3.2	40	50	+10
Tarea 1.3.3	25	20	-5
Tarea 1.3.4	40	50	+10
Tarea 1.4.1	10	5	-5
Tarea 1.4.2	10	5	-5
Tarea 1.5.1	20	25	+5
Tarea 1.5.2	50	55	+5
Tarea 1.5.3	20	20	0
Tarea 1.5.4	10	5	-5
Proyecto completo	**300**	**302**	**+2**

Tabla 3. Seguimiento del tiempo invertido en el proyecto.

Aunque durante el total del proyecto no se muestra una gran desviación, sí que existen en las diferentes partes pequeñas penalizaciones debido a que la implementación ha sido más costosa que lo planificado en inicialmente. Sin embargo, todo queda compensado a lo largo del resto de tareas del proyecto cuyo esfuerzo estimado se ha optimizado al máximo quedando corregido así el desfase inicial.

5.3. Calidad obtenida

La calidad del producto obtenido ha sido alta, ya que el software realiza su trabajo perfectamente. Además, ha sido mostrado a varios investigadores para que puedan familiarizarse con su uso (ya que la idea es que lo puedan comenzar a utilizar lo antes posible), mostrando todos ellos unas alta satisfacción y grandes expectativas. Todos concluyen con que esta utilidad será un beneficio en su trabajo diario. Además, los mecanismos de ocultación, el apartado de últimas subidas y su integración han sido unas mejoras que, aunque si constan en la realización de este proyecto, no eran parte de la idea inicial, con lo cual, la calidad total aumenta gracias a su implementación.

6. Lecciones Aprendidas

Aprender de quien no sabe

Intentar explicar un tema complejo a alguien que no tiene ni idea de lo que estás hablando es un reto bastante complicado. Y más si lo que intentas no es enseñárselo todo para que lo termine comprendiendo como si fueras un profesor, sino mostrarle una idea en la que estás trabajando.

Por más fácil que se lo intentes explicar, y más comparaciones que hagas, en muchos casos puede que lo que a ti te resulte muy sencillo, para los demás no, debido en gran medida a que posiblemente poseas los conocimientos para ello.

Sin embargo, esta práctica me ha resultado de gran utilidad. Cuando estás realizando un trabajo como lo es este proyecto final de carrera, lo más normal es que tu familia y amigos se preocupen por ti, y te pregunten qué estás haciendo, y cómo lo llevas. La segunda pregunta es fácil de responder, pero la primera, y más en un tema como la programación, en el que hay que tener muchos conocimientos previos en distintos temas, es muy difícil.

Aun así, el mero hecho de hacerlo te permite a ti mismo comprender de una mejor manera lo que estás haciendo, ya que tienes que reducir una montaña de conocimiento a arena y polvo, de tal manera que, al cambiar tu perspectiva, e intentar verlo de forma tan simple, te ayuda.

Usar un Sniffer de red

Desarrollar una aplicación web no es lo mismo que una aplicación de escritorio. En las segundas podemos buscar errores con mayor facilidad que en las primeras. Es por ello que necesitamos uso de herramientas extra cuando desarrollamos aplicaciones web. Una de ellas es el Sniffer, que permite capturar los paquetes de red que va recibiendo la máquina y observarlos.

Gracias a esta herramienta he podido arreglar errores en mi código y ver qué fallaba en determinados momentos, por lo que es muy recomendable su uso a cualquier desarrollador de aplicaciones que funcionen a través de cualquier mecanismo de red. En concreto, aunque sea muy simple, en este proyecto he usado el que viene incluido en el propio navegador.

Trabajar en una máquina virtual VS Trabajar fuera de ella

Cuando se está desarrollando una aplicación web que tendrá que funcionar por Internet, sin duda la mejor opción es trabajar con una máquina que actúe como servidor, y otra como cliente, y no hacerlo todo nunca dentro de la misma. Esto es fácil de conseguir usando por ejemplo una máquina virtual que actúe de servidor dentro de una única máquina.

Aunque desarrollar dentro de una única máquina sea más cómodo, la realidad es que la aplicación no terminará funcionando así, y por mucho que te esfuerces, seguramente terminarás cometiendo errores muy de novato por culpa de esto.

Por ejemplo, uno de los momentos en los que se hace más palpable este hecho (y el causante de que esta lección aprendida se encuentre en esta memoria) es cuando estás trabajando con rutas a ficheros. Cuando estás en una máquina todo es muy bonito, funciona a la primera, las rutas las construyes fácilmente y los permisos siempre son los adecuados. Sin embargo, si trabajas con dos, aparecen todos esos problemas que es necesario que resuelvas.

Modo DEBUG

Una aplicación web que funciona a través de un navegador normalmente depende de librerías y cargas que alargan el proceso de desarrollo cuando estás realizando pruebas.

Como programador tomar la decisión de crear un modo de depuración para la aplicación es algo esencial. En este proyecto he creado uno que simplemente muestra las variables fundamentales con lo que funciona el resto, y esto me ha permitido evitar los tiempos de carga innecesarios. El resultado es muy satisfactorio y útil, ya que aumenta drásticamente la eficiencia en el desarrollo.

7. Conclusiones del Proyecto

La realización de este Proyecto Fin de Grado ha supuesto la consecución de un reto personal por el tamaño, la complejidad y los conocimientos sobre bioinformática previos. En concreto, al no disponer de demasiada información sobre dicho campo, existía el temor de no conseguir llegar a buen puerto. Sin embargo, los conocimientos se han ido adquiriendo poco a poco y en este sentido no ha habido problemas.

Además, también se ha utilizado un lenguaje no conocido como PHP, al que se le ha tenido que dedicar un tiempo para aprender a manejarlo, al igual que Drupal, el sistema gestor de contenidos usado en el proyecto. Ni siquiera sabía de su existencia, y al completar el proyecto la preparación es bastante amplia.

A nivel profesional, los resultados han sido excelentes, y la gestión de un proyecto de este tamaño hace que me pueda sentir mucho más preparado de cara tanto a nuevos proyectos, como a enfrentarme al mundo laboral.

Para terminar, decir que ha sido una gran experiencia, y que me ha quitado el miedo a programar en ámbitos que puedan parecer muy complejos desde fuera, pero que desde dentro sigue siendo el código fuente de siempre.

8. Bibliografía

- Fundamentos de Biología Molecular. Dorcas, J. Orengo Ferriz (2013).
- Genómica Computacional. Enrique Blanco García (2013).
- Documentación de *MATE*: https://goo.gl/xlYlpy
- Documentación de *IGV.js*: https://goo.gl/Ez58z1
- Documentación de Drupal: https://goo.gl/XPssLa

9. Anexos

- **Anexo 1.** Explicación del script GenomicExplorer.
- **Anexo 2.** Manual de uso de la aplicación.